Table of Contents

What is Azure Scheduler?

6/27/2017 • 1 min to read • Edit Online

Azure Scheduler allows you to declaratively describe actions to run in the cloud. It then schedules and runs those actions automatically. Scheduler does this by using the Azure portal, code, REST API, or Azure PowerShell.

Scheduler creates, maintains, and invokes scheduled work. Scheduler does not host any workloads or run any code. It only *invokes* code hosted elsewhere—in Azure, on-premises, or with another provider. It invokes via HTTP, HTTPS, a storage queue, a service bus queue, or a service bus topic.

Scheduler schedules jobs, keeps a history of job execution results that one can review, and deterministically and reliably schedules workloads to be run. Azure WebJobs (part of the Web Apps feature in Azure App Service) and other Azure scheduling capabilities use Scheduler in the background. The Scheduler REST API helps manage the communication for these actions. As such, Scheduler supports complex schedules and advanced recurrence easily.

There are several scenarios that lend themselves to the usage of Scheduler. For example:

- *Recurring application actions:* Periodically gathering data from Twitter into a feed.
- *Daily maintenance:* Daily pruning of logs, performing backups, and other maintenance tasks. For example, an administrator may choose to back up the database at 1:00 A.M. every day for the next nine months.

Scheduler allows you to create, update, delete, view, and manage jobs and job collections programmatically, by using scripts, and in the portal.

See also

Azure Scheduler concepts, terminology, and entity hierarchy

Get started using Scheduler in the Azure portal

Plans and billing in Azure Scheduler

How to build complex schedules and advanced recurrence with Azure Scheduler

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler high-availability and reliability

Azure Scheduler limits, defaults, and error codes

Azure Scheduler outbound authentication

Get started with Azure Scheduler in Azure portal

6/27/2017 • 2 min to read • Edit Online

It's easy to create scheduled jobs in Azure Scheduler. In this tutorial, you'll learn how to create a job. You'll also learn Scheduler's monitoring and management capabilities.

Create a job

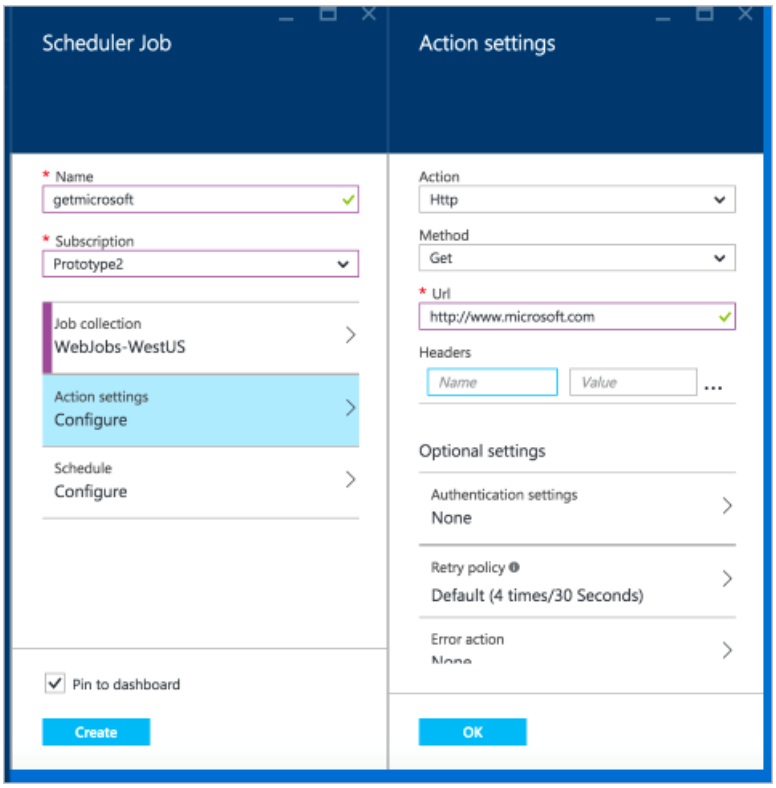
1. Sign in to Azure portal.
2. Click **+New** > type *Scheduler* in the search box > select **Scheduler** in results > click **Create**.

3. Let's create a job that simply hits http://www.microsoft.com/ with a GET request. In the **Scheduler Job** screen, enter the following information:

   a. **Name:** `getmicrosoft`
   b. **Subscription:** Your Azure subscription
   c. **Job Collection:** Select an existing job collection, or click **Create New** > enter a name.

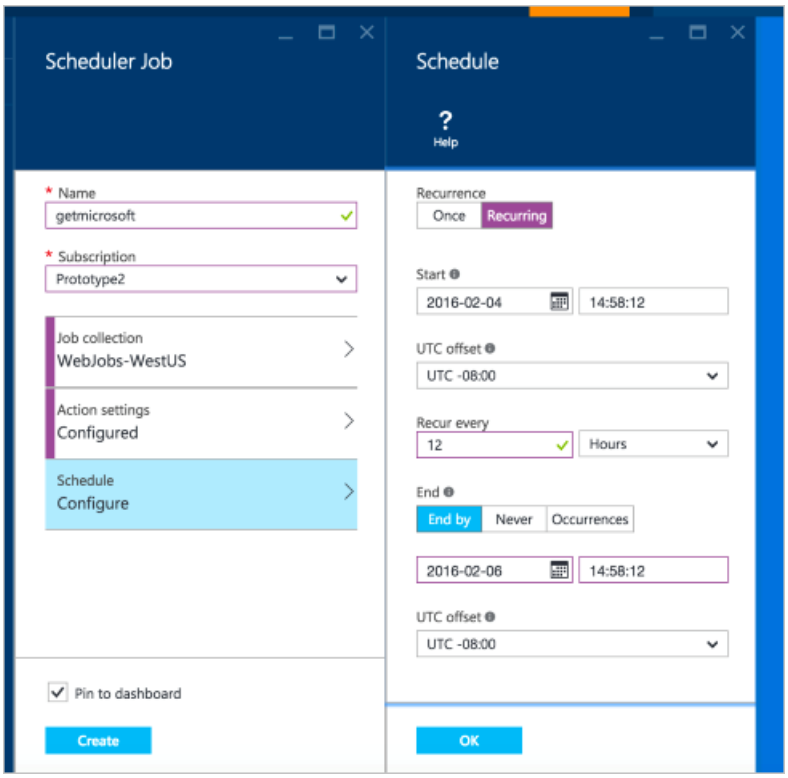
4. Next, in **Action Settings**, define the following values:

   a. **Action Type:** `HTTP`
   b. **Method:** `GET`
   c. **URL:** `http://www.microsoft.com`



5. Finally, let's define a schedule. The job could be defined as a one-time job, but let's pick a recurrence schedule:

   a. **Recurrence**: `Recurring`
   b. **Start**: Today's date
   c. **Recur every**: `12 Hours`
   d. **End by**: Two days from today's date

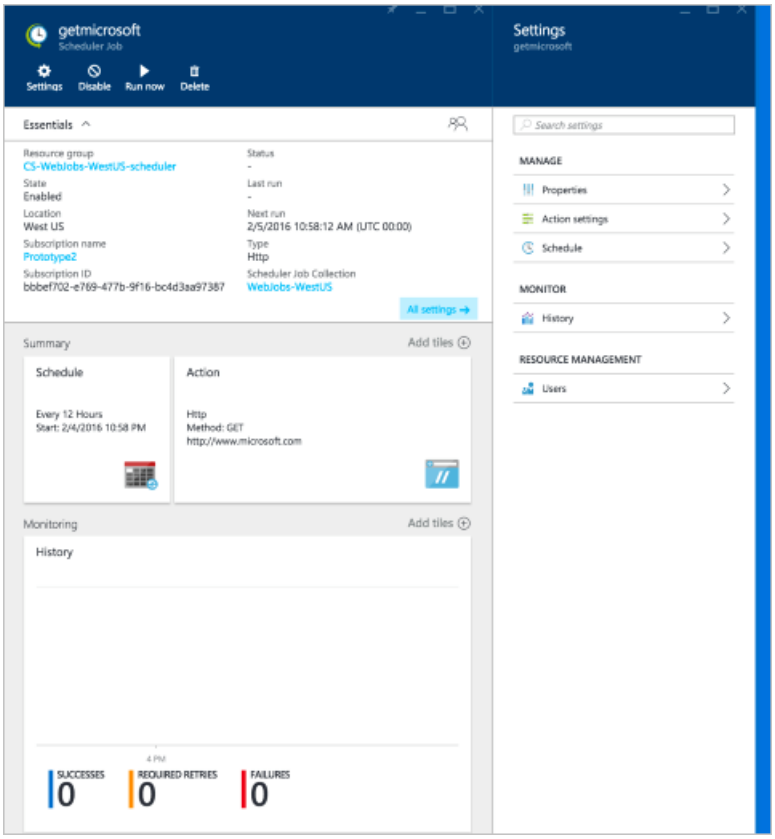


6. Click **Create**

Manage and monitor jobs

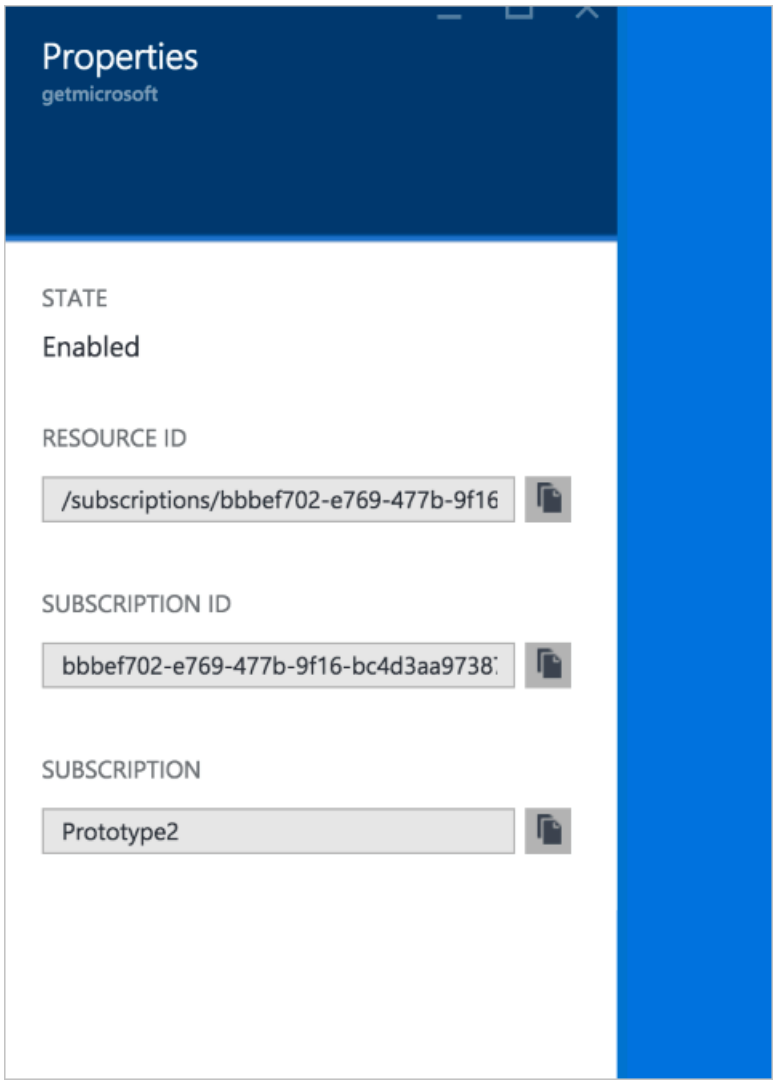
Once a job is created, it appears in the main Azure dashboard. Click the job and a new window opens with the following tabs:

1. Properties
2. Action Settings
3. Schedule
4. History
5. Users



Properties

These read-only properties describe the management metadata for the Scheduler job.



Action settings

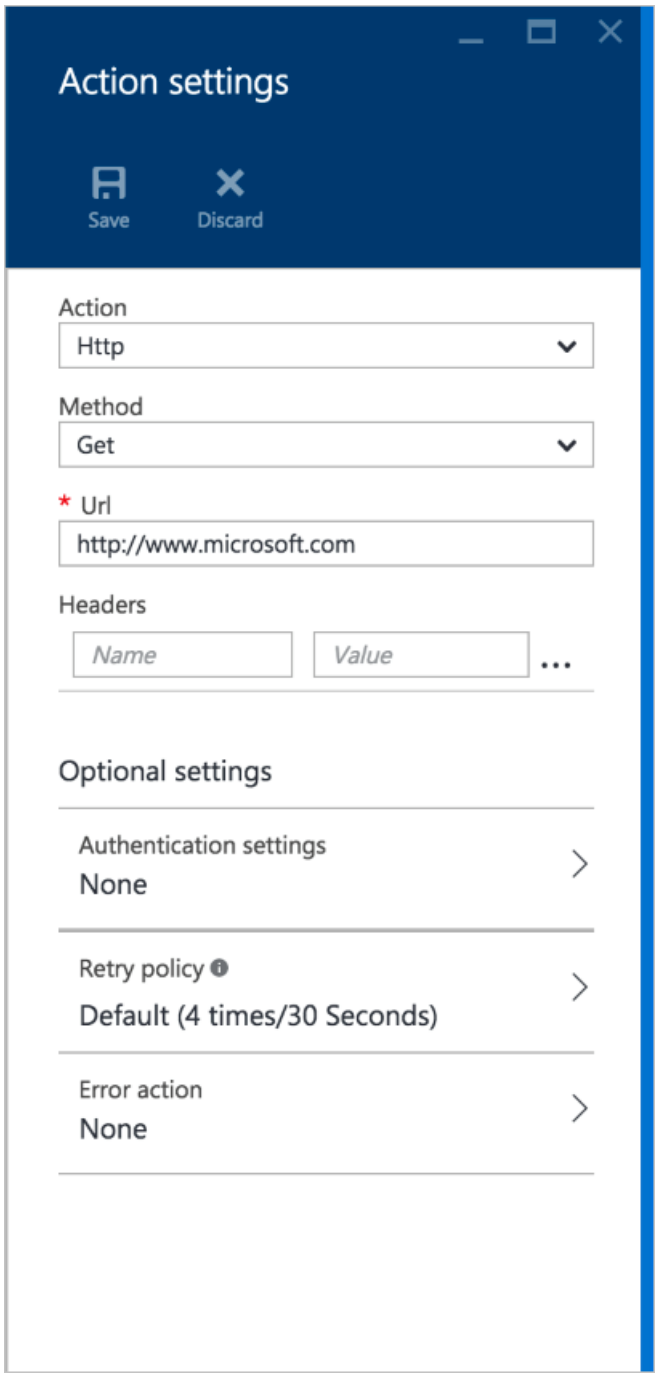
Clicking on a job in the **Jobs** screen allows you to configure that job. This lets you configure advanced settings, if you didn't configure them in the quick-create wizard.

For all action types, you may change the retry policy and the error action.

For HTTP and HTTPS job action types, you may change the method to any allowed HTTP verb. You may also add, delete, or change the headers and basic authentication information.

For storage queue action types, you may change the storage account, queue name, SAS token, and body.

For service bus action types, you may change the namespace, topic/queue path, authentication settings, transport type, message properties, and message body.

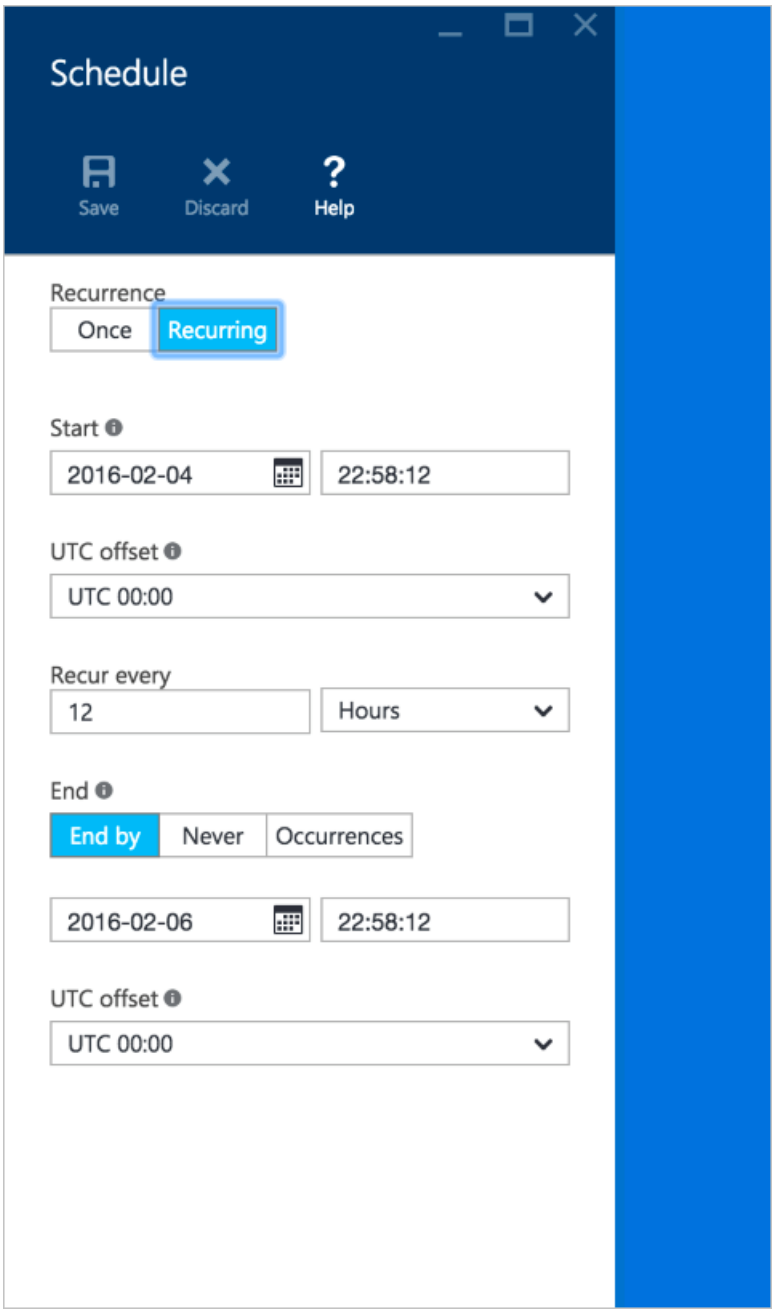


Schedule

This lets you reconfigure the schedule, if you'd like to change the schedule you created in the quick-create wizard.

This is an opportunity to build complex schedules and advanced recurrence in your job

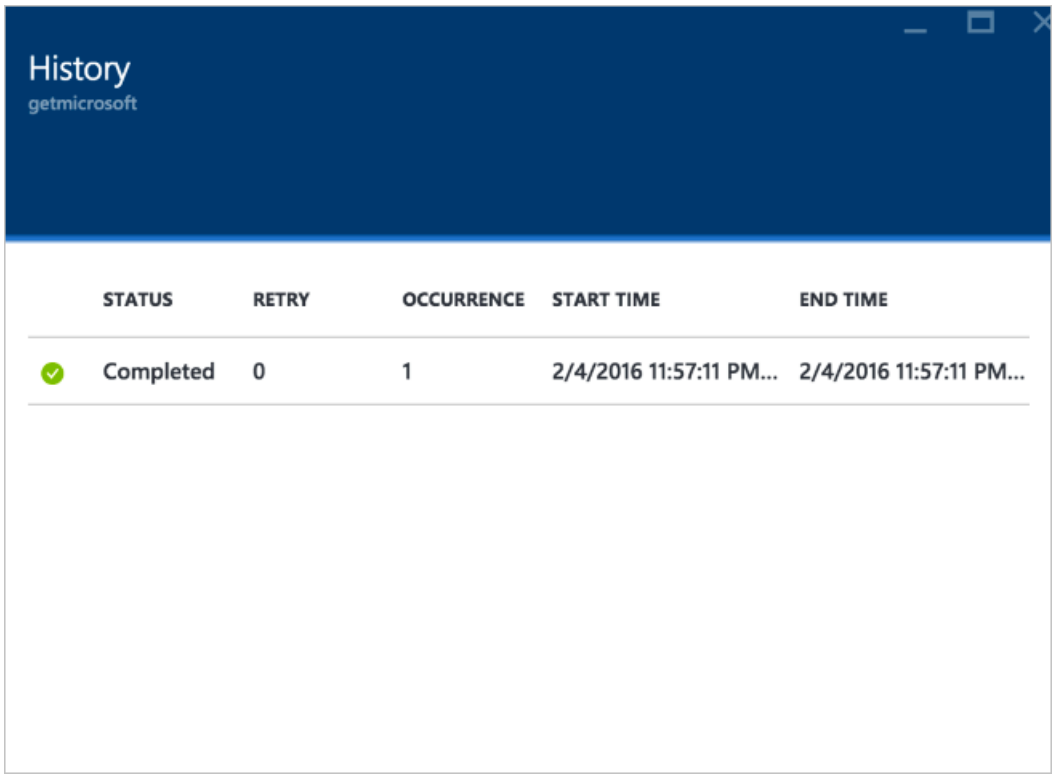
You may change the start date and time, recurrence schedule, and the end date and time (if the job is recurring.)



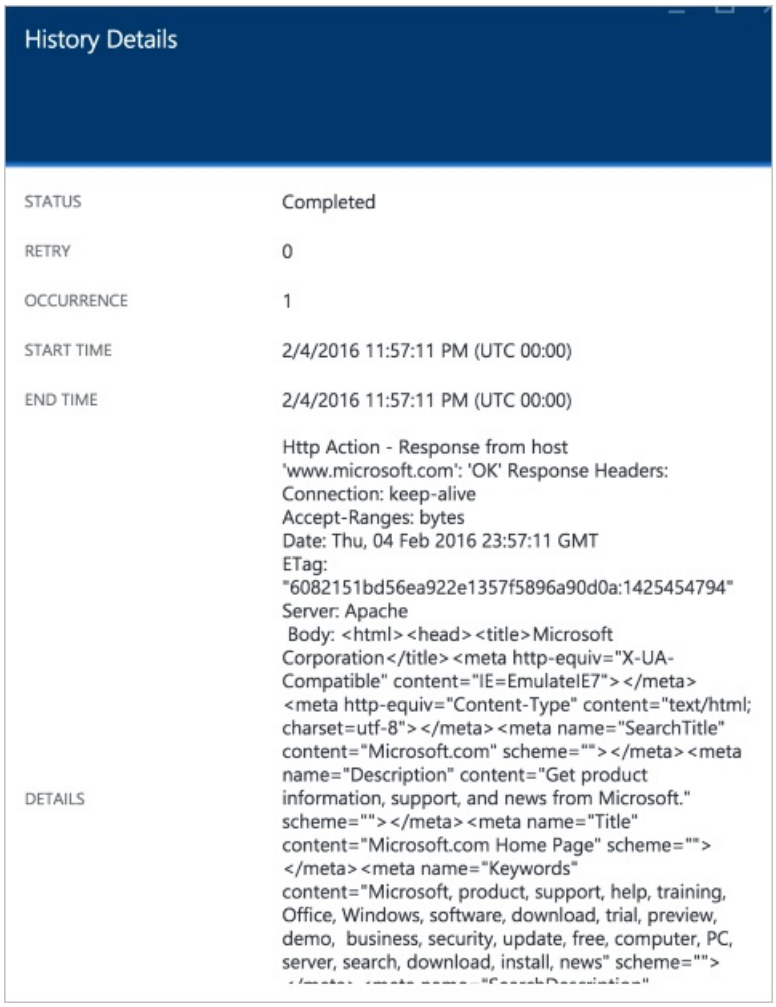
History

The **History** tab displays selected metrics for every job execution in the system for the selected job. These metrics provide real-time values regarding the health of your Scheduler:

1. Status
2. Details
3. Retry attempts
4. Occurrence: 1st, 2nd, 3rd, etc.
5. Start time of execution
6. End time of execution



You can click on a run to view its **History Details**, including the whole response for every execution. This dialog box also allows you to copy the response to the clipboard.



Users

Azure Role-Based Access Control (RBAC) enables fine-grained access management for Azure Scheduler. To learn how to use the Users tab, refer to Azure Role-Based Access Control

See also

What is Scheduler?

Scheduler concepts, terminology, and entity hierarchy

Plans and billing in Azure Scheduler

How to build complex schedules and advanced recurrence with Azure Scheduler

Scheduler REST API reference

Scheduler PowerShell cmdlets reference

Scheduler high-availability and reliability

Scheduler limits, defaults, and error codes

Scheduler outbound authentication

Scheduler concepts, terminology, + entity hierarchy

6/27/2017 • 7 min to read • Edit Online

Scheduler entity hierarchy

The following table describes the main resources exposed or used by the Scheduler API:

RESOURCE	DESCRIPTION
Job collection	A job collection contains a group of jobs and maintains settings, quotas, and throttles that are shared by jobs within the collection. A job collection is created by a subscription owner and groups jobs together based on usage or application boundaries. It's constrained to one region. It also allows the enforcement of quotas to constrain the usage of all jobs in that collection. The quotas include MaxJobs and MaxRecurrence.
Job	A job defines a single recurrent action, with simple or complex strategies for execution. Actions may include HTTP, storage queue, service bus queue, or service bus topic requests.
Job history	A job history represents details for an execution of a job. It contains success vs. failure, as well as any response details.

Scheduler entity management

At a high level, the scheduler and the service management API expose the following operations on the resources:

CAPABILITY	DESCRIPTION AND URI ADDRESS
Job collection management	GET, PUT, and DELETE support for creating and modifying job collections and the jobs contained therein. A job collection is a container for jobs and maps to quotas and shared settings. Examples of quotas, described later, are maximum number of jobs and smallest recurrence interval. PUT and DELETE: `https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/prov` GET: `https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/prov`
Job management	GET, PUT, POST, PATCH, and DELETE support for creating and modifying jobs. All jobs must belong to a job collection that already exists, so there is no implicit creation. `https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/prov`
Job history management	GET support for fetching 60 days of job execution history, such as job elapsed time and job execution results. Adds query string parameter support for filtering based on state and status. `https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/prov`

Job types

There are multiple types of jobs: HTTP jobs (including HTTPS jobs that support SSL), storage queue jobs, service bus queue jobs, and service bus topic jobs. HTTP jobs are ideal if you have an endpoint of an existing workload or service. You can use storage queue jobs to post messages to storage queues, so those jobs are ideal for workloads that use storage queues. Similarly, service bus jobs are ideal for workloads that use service bus queues and topics.

The "job" entity in detail

At a basic level, a scheduled job has several parts:

- The action to perform when the job timer fires
- (Optional) The time to run the job
- (Optional) When and how often to repeat the job
- (Optional) An action to fire if the primary action fails

Internally, a scheduled job also contains system-provided data such as the next scheduled execution time.

The following code provides a comprehensive example of a scheduled job. Details are provided in subsequent sections.

```
{
    "startTime": "2012-08-04T00:00Z",              // optional
    "action":
    {
        "type": "http",
        "retryPolicy": { "retryType":"none" },
        "request":
        {
            "uri": "http://contoso.com/foo",        // required
            "method": "PUT",                        // required
            "body": "Posting from a timer",         // optional
            "headers":                              // optional

            {
                "Content-Type": "application/json"
            },
        },
       "errorAction":
       {
           "type": "http",
           "request":
           {
               "uri": "http://contoso.com/notifyError",
               "method": "POST",
           },
       },
    },
    "recurrence":                                   // optional
    {
        "frequency": "week",                        // can be "year" "month" "day" "week" "minute"
        "interval": 1,                              // optional, how often to fire (default to 1)
        "schedule":                                 // optional (advanced scheduling specifics)
        {
            "weekDays": ["monday", "wednesday", "friday"],
            "hours": [10, 22]
        },
        "count": 10,                                 // optional (default to recur infinitely)
        "endTime": "2012-11-04",                     // optional (default to recur infinitely)
    },
    "state": "disabled",                           // enabled or disabled
    "status":                                       // controlled by Scheduler service
    {
        "lastExecutionTime": "2007-03-01T13:00:00Z",
        "nextExecutionTime": "2007-03-01T14:00:00Z ",
        "executionCount": 3,
                                          "failureCount": 0,
                                          "faultedCount": 0
    },
}
```

As seen in the sample scheduled job above, a job definition has several parts:

- Start time (“startTime”)
- Action (“action”), which includes error action (“errorAction”)
- Recurrence (“recurrence”)
- State (“state”)
- Status (“status”)
- Retry policy (“retryPolicy”)

Let’s examine each of these in detail:

startTime

The "startTime” is the start time and allows the caller to specify a time zone offset on the wire in ISO-8601 format.

action and errorAction

The “action” is the action invoked on each occurrence and describes a type of service invocation. The action is what will be executed on the provided schedule. Scheduler supports HTTP, storage queue, service bus topic, and service bus queue actions.

The action in the example above is an HTTP action. Below is an example of a storage queue action:

```
{
        "type": "storageQueue",
        "queueMessage":
        {
            "storageAccount": "myStorageAccount",  // required
            "queueName": "myqueue",                // required
            "sasToken": "TOKEN",                   // required
            "message":                             // required
                "My message body",
        },
}
```

Below is an example of a service bus topic action.

"action": { "type": "serviceBusTopic", "serviceBusTopicMessage": { "topicPath": "t1",
"namespace": "mySBNamespace", "transportType": "netMessaging", // Can be either netMessaging or AMQP "authentication": { "sasKeyName": "QPolicy", "type":
"sharedAccessKey" }, "message": "Some message", "brokeredMessageProperties": {}, "customMessageProperties": { "appname": "FromScheduler" } }, }

Below is an example of a service bus queue action:

"action": { "serviceBusQueueMessage": { "queueName": "q1",
"namespace": "mySBNamespace", "transportType": "netMessaging", // Can be either netMessaging or AMQP "authentication": {
"sasKeyName": "QPolicy", "type": "sharedAccessKey" }, "message": "Some message",
"brokeredMessageProperties": {}, "customMessageProperties": { "appname": "FromScheduler" } }, "type": "serviceBusQueue" }

The “errorAction” is the error handler, the action invoked when the primary action fails. You can use this variable to call an error-handling endpoint or send a user notification. This can be used for reaching a secondary endpoint in the case that the primary is not available (e.g., in the case of a disaster at the endpoint’s site) or can be used for notifying

an error handling endpoint. Just like the primary action, the error action can be simple or composite logic based on other actions. To learn how to create a SAS token, refer to Create and Use a Shared Access Signature.

recurrence

Recurrence has several parts:

- Frequency: One of minute, hour, day, week, month, year
- Interval: Interval at the given frequency for the recurrence
- Prescribed schedule: Specify minutes, hours, weekdays, months, and monthdays of the recurrence
- Count: Count of occurrences
- End time: No jobs will execute after the specified end time

A job is recurring if it has a recurring object specified in its JSON definition. If both count and endTime are specified, the completion rule that occurs first is honored.

state

The state of the job is one of four values: enabled, disabled, completed, or faulted. You can PUT or PATCH jobs so as to update them to the enabled or disabled state. If a job has been completed or faulted, that is a final state that cannot be updated (though the job can still be DELETED). An example of the state property is as follows:

```
    "state": "disabled", // enabled, disabled, completed, or faulted
```

Completed and faulted jobs are deleted after 60 days.

status

Once a Scheduler job has started, information will be returned about the current status of the job. This object is not settable by the user—it's set by the system. However, it is included in the job object (rather than a separate linked resource) so that one can obtain the status of a job easily.

Job status includes the time of the previous execution (if any), the time of the next scheduled execution (for in-progress jobs), and the execution count of the job.

retryPolicy

If a Scheduler job fails, it is possible to specify a retry policy to determine whether and how the action is retried. This is determined by the **retryType** object—it is set to **none** if there is no retry policy, as shown above. Set it to **fixed** if there is a retry policy.

To set a retry policy, two additional settings may be specified: a retry interval (**retryInterval**) and the number of retries (**retryCount**).

The retry interval, specified with the **retryInterval** object, is the interval between retries. Its default value is 30 seconds, its minimum configurable value is 15 seconds, and its maximum value is 18 months. Jobs in Free job collections have a minimum configurable value of 1 hour. It is defined in the ISO 8601 format. Similarly, the value of the number of retries is specified with the **retryCount** object; it is the number of times a retry is attempted. Its default value is 4, and its maximum value is 20. Both **retryInterval** and **retryCount** are optional. They are given their default values if **retryType** is set to **fixed** and no values are specified explicitly.

See also

What is Scheduler?

Get started using Scheduler in the Azure portal

Plans and billing in Azure Scheduler

How to build complex schedules and advanced recurrence with Azure Scheduler

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler high-availability and reliability

Azure Scheduler limits, defaults, and error codes

Azure Scheduler outbound authentication

Plans and Billing in Azure Scheduler

6/27/2017 • 4 min to read • Edit Online

Job Collection Plans

Job collections are the billable entity in Azure Scheduler. Job collections contain a number of jobs and come in three plans – Free, Standard, and Premium – that are described below.

JOB COLLECTION PLAN	MAX # OF JOBS PER JOB COLLECTION	MAX RECURRENCE	MAX JOB COLLECTIONS PER SUBSCRIPTION	LIMITS
Free	5 jobs per job collection	Once per hour. Cannot execute jobs more often than once an hour	A subscription is allowed up to 1 free job collection	Cannot use HTTP outbound authorization object
Standard	50 jobs per job collection	Once per minute. Cannot execute jobs more often than once a minute	A subscription is allowed up to 100 standard job collections	Access to full feature set of Scheduler
P10 Premium	50 jobs per job collection	Once per minute. Cannot execute jobs more often than once a minute	A subscription is allowed up to 10,000 P10 Premium job collections. Contact us for more.	Access to full feature set of Scheduler
P20 Premium	1000 jobs per job collection	Once per minute. Cannot execute jobs more often than once a minute	A subscription is allowed up to 10,000 P20 Premium job collections. Contact us for more.	Access to full feature set of Scheduler

Upgrades and Downgrades of Job Collection Plans

You may upgrade or downgrade a job collection plan anytime among the Free, Standard, and Premium plans. However, when downgrading to a free job collection, the downgrade may fail for one of the following reasons:

- A free job collection already exists in the subscription
- A job in the job collection has a higher recurrence than allowed for jobs in free job collections. The maximum recurrence allowed in a free job collection is once per hour
- There are more than 5 jobs in the job collection
- A job in the job collection has an HTTP or HTTPS action that uses an HTTP outbound authorization object

Billing and Azure Plans

Subscriptions are not charged for free job collections. If you have more than 100 standard job collections (10 standard billing units), then it's a better deal to have all job collections in the premium plan.

If you have one standard job collection and one premium job collection, you are billed one standard billing unit *and* one premium billing unit. The Scheduler service bills based on the number of active job collections that are set to either standard or premium; this is explained further in the next two sections.

Standard Billable Units

A standard billable unit can include up to 10 standard job collections. Since a standard job collection can have up to 50 jobs per job collection, one standard billing unit allows a subscription to have up to 500 jobs – up to almost 22 million job executions per month.

If you have between 1 and 10 standard job collections, you'll be billed for 1 standard billing unit. If you have between 11 and 20 standard job collections, you'll be billed for 2 standard billing units. If you have between 21 and 30 standard job collections, you'll be billed for 3 standard billing units, and so on.

P10 Premium Billable Units

A P10 premium billable unit can include up to 10,000 P10 premium job collections. Since a P10 premium job collection can have up to 50 jobs per job collection, one premium billing unit allows a subscription to have up to 500,000 jobs – up to almost 22 billion job executions per month.

If you have between 1 and 10,000 premium job collections, you'll be billed for 1 P10 premium billing unit. If you have between 10,001 and 20,000 premium job collections, you'll be billed for 2 P10 premium billing units, and so on.

Thus, P10 premium job collections have the same functionality as the standard job collections but provide a price break in case your application requires a lot of job collections.

P20 Premium Billable Units

A P20 premium billable unit can include up to 5,000 P20 premium job collections. Since a P20 premium job collection can have up to 1,000 jobs per job collection, one premium billing unit allows a subscription to have up to 5,000,000 jobs – up to almost 220 billion job executions per month.

P20 premium job collections provides the same capabilities as P10 premium job collections but also supports a greater number jobs per job collection and a greater total number of jobs overall than P10 premium allowing you to have more scalability.

Billing and Active Status

Job collections are always active unless your entire subscription has gone into some temporary disabled state due to billing issues. The only way to ensure that a job collection is not billed is to either set it to the *Free* plan or to delete the job collection.

Although you may disable all jobs within a job collection in a single operation, it does not change the billing status of the job collection – the job collection will *still* be billed. Similarly, empty job collections are considered active and will be billed.

Pricing

For pricing details, please see Scheduler Pricing.

See Also

What is Scheduler?

Azure Scheduler concepts, terminology, and entity hierarchy

Get started using Scheduler in the Azure portal

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler high-availability and reliability

Azure Scheduler limits, defaults, and error codes

Azure Scheduler outbound authentication

Scheduler Limits and Defaults

6/27/2017 • 2 min to read • Edit Online

Scheduler Quotas, Limits, Defaults, and Throttles

The following table describes each of the major quotas, limits, defaults, and throttles in Azure Scheduler.

RESOURCE	LIMIT DESCRIPTION
Job size	Maximum job size is 16K. If a PUT or a PATCH results in a job larger than these limits, a 400 Bad Request status code is returned.
Request URL size	Maximum size of the request URL is 2048 chars.
Aggregate header size	Maximum aggregate header size is 4096 chars.
Header count	Maximum header count is 50 headers.
Body size	Maximum body size is 8192 chars.
Recurrence span	Maximum recurrence span is 18 months.
Time to start time	Maximum "time to start time" is 18 months.
Job history	Maximum response body stored in job history is 2048 bytes.
Frequency	The default max frequency quota is 1 hour in a free job collection and 1 minute in a standard job collection. The max frequency is configurable on a job collection to be lower than the maximum. All jobs in the job collection are limited the value set on the job collection. If you attempt to create a job with a higher frequency than the maximum frequency on the job collection then request will fail with a 409 Conflict status code.
Jobs	The default max jobs quota is 5 jobs in a free job collection and 50 jobs in a standard job collection. The maximum number of jobs is configurable on a job collection. All jobs in the job collection are limited the value set on the job collection. If you attempt to create more jobs than the maximum jobs quota, then the request fails with a 409 Conflict status code.
Job collections	Maximum number of job collection per subscription is 200,000.
Job history retention	Job history is retained for up to 2 months or up to the last 1000 executions.
Completed and faulted job retention	Completed and faulted jobs are retained for 60 days.

RESOURCE	LIMIT DESCRIPTION
Timeout	There's a static (not configurable) request timeout of 60 seconds for HTTP actions. For longer running operations, follow HTTP asynchronous protocols; for example, return a 202 immediately but continue working in the background.

The x-ms-request-id Header

Every request made against the Scheduler service returns a response header named**x-ms-request-id**. This header contains an opaque value that uniquely identifies the request.

If a request is consistently failing and you have verified that the request is properly formulated, you may use this value to report the error to Microsoft. In your report, include the value of x-ms-request-id, the approximate time that the request was made, the identifier of the subscription, job collection, and/or job, and the type of operation that the request attempted.

See Also

What is Scheduler?

Azure Scheduler concepts, terminology, and entity hierarchy

Get started using Scheduler in the Azure portal

Plans and billing in Azure Scheduler

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler high-availability and reliability

Azure Scheduler outbound authentication

Scheduler High-Availability and Reliability

6/27/2017 • 3 min to read • Edit Online

Azure Scheduler High-Availability

As a core Azure platform service, Azure Scheduler is highly available and features both geo-redundant service deployment and geo-regional job replication.

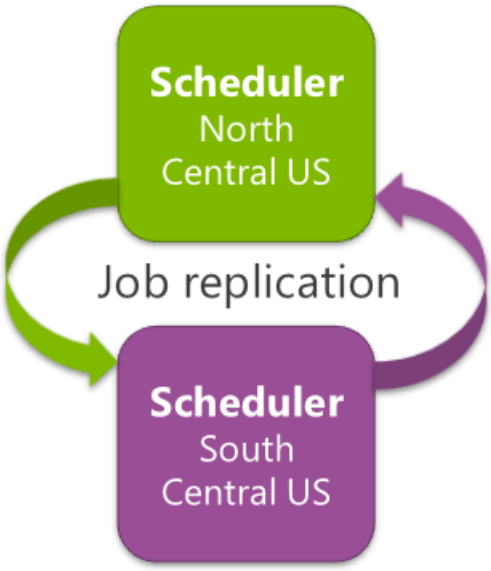
Geo-redundant service deployment

Azure Scheduler is available via the UI in almost every geo region that's in Azure today. The list of regions that Azure Scheduler is available in is listed here. If a data center in a hosted region is rendered unavailable, the failover capabilities of Azure Scheduler are such that the service is available from another data center.

Geo-regional job replication

Not only is the Azure Scheduler front-end available for management requests, but your own job is also geo-replicated. When there's an outage in one region, Azure Scheduler fails over and ensures that the job is run from another data center in the paired geographic region.

For example, if you've created a job in South Central US, Azure Scheduler automatically replicates that job in North Central US. When there's a failure in South Central US, Azure Scheduler ensures that the job is run from North Central US.



As a result, Azure Scheduler ensures that your data stays within the same broader geographic region in case of an Azure failure. As a result, you need not duplicate your job just to add high availability – Azure Scheduler automatically provides high-availability capabilities for your jobs.

Azure Scheduler Reliability

Azure Scheduler guarantees its own high-availability and takes a different approach to user-created jobs. For example, your job may invoke an HTTP endpoint that's unavailable. Azure Scheduler nonetheless tries to execute your job successfully, by giving you alternative options to deal with failure. Azure Scheduler does this in two ways:

Configurable Retry Policy via "retryPolicy"

Azure Scheduler allows you to configure a retry policy. By default, if a job fails, Scheduler tries the job again four more times, at 30-second intervals. You may re-configure this retry policy to be more aggressive (for example, ten times, at 30-second intervals) or looser (for example, two times, at daily intervals.)

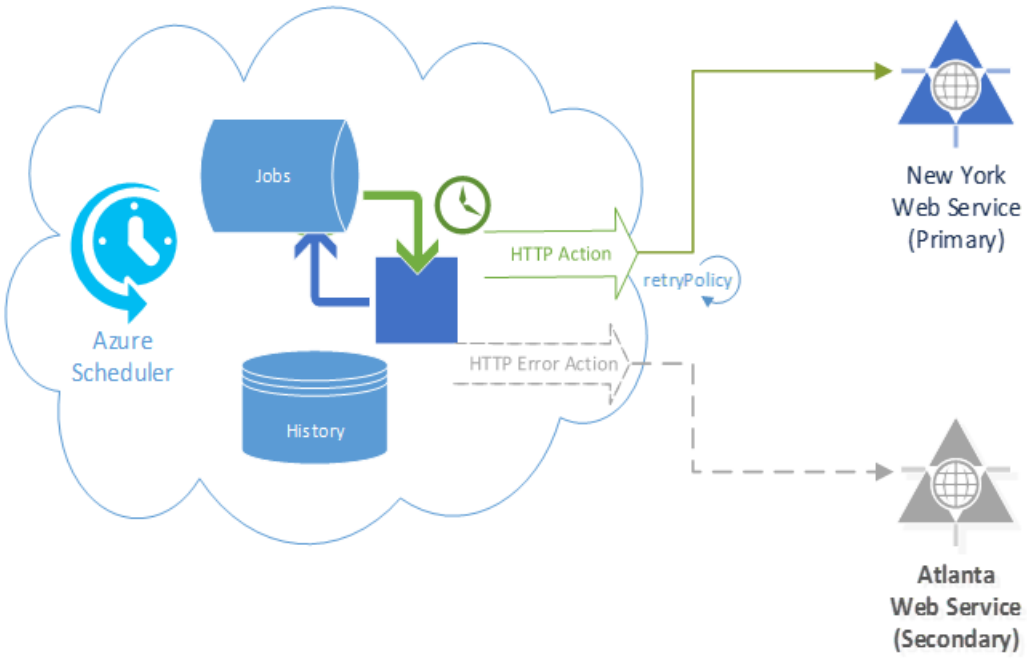
As an example of when this may help, you may create a job that runs once a week and invokes an HTTP endpoint. If the HTTP endpoint is down for a few hours when your job runs, you may not want to wait one more week for the job to run again since even the default retry policy will fail. In such cases, you may reconfigure the standard retry policy to retry every three hours (for example) instead of every 30 seconds.

To learn how to configure a retry policy, refer to retryPolicy.

Alternate Endpoint Configurability via "errorAction"

If the target endpoint for your Azure Scheduler job remains unreachable, Azure Scheduler falls back to the alternate error-handling endpoint after following its retry policy. If an alternate error-handling endpoint is configured, Azure Scheduler invokes it. With an alternate endpoint, your own jobs are highly available in the face of failure.

As an example, in the diagram below, Azure Scheduler follows its retry policy to hit a New York web service. After the retries fail, it checks if there's an alternate. It then goes ahead and starts making requests to the alternate with the same retry policy.



Note that the same retry policy applies to both the original action and the alternate error action. It's also possible to have the alternate error action's action type be different from the main action's action type. For example, while the main action may be invoking an HTTP endpoint, the error action may instead be a storage queue, service bus queue, or service bus topic action that does error-logging.

To learn how to configure an alternate endpoint, refer to errorAction.

See Also

What is Scheduler?

Azure Scheduler concepts, terminology, and entity hierarchy

Get started using Scheduler in the Azure portal

Plans and billing in Azure Scheduler

How to build complex schedules and advanced recurrence with Azure Scheduler

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler limits, defaults, and error codes

Azure Scheduler outbound authentication

How to Build Complex Schedules and Advanced Recurrence with Azure Scheduler

6/27/2017 • 12 min to read • Edit Online

Overview

At the heart of an Azure Scheduler job is the *schedule*. The schedule determines when and how the Scheduler executes the job.

Azure Scheduler allows you to specify different one-time and recurring schedules for a job. *One-time* schedules fire once at a specified time – effectively, they are *recurring* schedules that execute only once. Recurring schedules fire on a predetermined frequency.

With this flexibility, Azure Scheduler lets you support a wide variety of business scenarios:

- Periodic data cleanup – e.g., every day, delete all tweets older than 3 months
- Archival – e.g., every month, push invoice history to backup service
- Requests for external data – e.g., every 15 minutes, pull new ski weather report from NOAA
- Image processing – e.g. every weekday, during off-peak hours, use cloud computing to compress images uploaded that day

In this article, we walk through example jobs that you can create with Azure Scheduler. We provide the JSON data that describes each schedule. If you use the Scheduler REST API, you can use this same JSON for creating an Azure Scheduler job.

Supported Scenarios

The many examples in this topic illustrate the breadth of scenarios that Azure Scheduler supports. Broadly, these examples illustrate how to create schedules for many behavior patterns, including the ones below:

- Run once at a particular date and time
- Run and recur a number of explicit times
- Run immediately and recur
- Run and recur every *n* minutes, hours, days, weeks, or months, starting at a particular time
- Run and recur at weekly or monthly frequency but only on specific days, specific days of week, or specific days of month
- Run and recur at multiple times in a period – e.g., last Friday and Monday of every month, or at 5:15am and 5:15pm every day

Dates and DateTimes

Dates in Azure Scheduler jobs follow the ISO-8601 specification and include only the date.

Date-Time references in Azure Scheduler jobs follow the ISO-8601 specification and include both date and time parts. A Date-Time that does not specify a UTC offset is assumed to be UTC.

How To: Use JSON and REST API for Creating Schedules

To create a simple schedule using the Azure Scheduler REST API, first register your subscription with a resource provider (the provider name for Scheduler is *Microsoft.Scheduler*), then create a job collection, and finally create a

job. When you create a job, you can specify scheduling and recurrence using JSON like the one excerpted below:

```
{
    "startTime": "2012-08-04T00:00Z", // optional
     …
    "recurrence":                     // optional
    {
        "frequency": "week",     // can be "year" "month" "day" "week" "hour" "minute"
        "interval": 1,               // optional, how often to fire (default to 1)
        "schedule":                  // optional (advanced scheduling specifics)
        {
            "weekDays": ["monday", "wednesday", "friday"],
            "hours": [10, 22]
        },
        "count": 10,                 // optional (default to recur infinitely)
        "endTime": "2012-11-04",     // optional (default to recur infinitely)
    },
    …
}
```

Overview: Job Schema Basics

The following table provides a high-level overview of the major elements related to recurrence and scheduling in a job:

JSON NAME	DESCRIPTION
startTime	*startTime* is a Date-Time. For simple schedules, *startTime* is the first occurrence and for complex schedules, the job will start no sooner than *startTime*.
recurrence	The *recurrence* object specifies recurrence rules for the job and the recurrence the job will execute with. The recurrence object supports the elements *frequency, interval, endTime, count,* and *schedule*. If *recurrence* is defined, *frequency* is required; the other elements of *recurrence* are optional.
frequency	The *frequency* string representing the frequency unit at which the job recurs. Supported values are "*minute*", "*hour*", "*day*", "*week*", or "*month*."
interval	The *interval* is a positive integer and denotes the interval for the *frequency* that determines how often the job will run. For example, if *interval* is 3 and *frequency* is "week", the job recurs every 3 weeks. Azure Scheduler supports a maximum *interval* of 18 months for monthly frequency, 78 weeks for weekly frequency, or 548 days for daily frequency. For hour and minute frequency, the supported range is 1 <= *interval* <= 1000.
endTime	The *endTime* string specifies the date-time past which the job should not execute. It is not valid to have an *endTime* in the past. If no *endTime* or count is specified, the job runs infinitely. Both *endTime* and *count* cannot be included for the same job.

JSON NAME	DESCRIPTION
count	The *count* is a positive integer (greater than zero) that specifies the number of times this job should run before completing. The *count* represents the number of times the job runs before being determined as completed. For example, for a job that is executed daily with *count* 5 and start date of Monday, the job completes after execution on Friday. If the start date is in the past, the first execution is calculated from the creation time. If no *endTime* or *count* is specified, the job runs infinitely. Both *endTime* and *count* cannot be included for the same job.
schedule	A job with a specified frequency alters its recurrence based on a recurrence schedule. A *schedule* contains modifications based on minutes, hours, week days, month days, and week number.

Overview: Job Schema Defaults, Limits, and Examples

After this overview, let's discuss each of these elements in detail.

JSON NAME	VALUE TYPE	REQUIRED?	DEFAULT VALUE	VALID VALUES	EXAMPLE
startTime	String	No	None	ISO-8601 Date-Times	`"startTime" : "2013-01-09T09:30:00-08:00"`
recurrence	Object	No	None	Recurrence object	`"recurrence" : { "frequency" : "monthly", "interval" : 1 }`
frequency	String	Yes	None	"minute", "hour", "day", "week", "month"	`"frequency" : "hour"`
interval	Number	No	1	1 to 1000.	`"interval":10`
endTime	String	No	None	Date-Time value representing a time in the future	`"endTime" : "2013-02-09T09:30:00-08:00"`
count	Number	No	None	>= 1	`"count": 5`
schedule	Object	No	None	Schedule object	`"schedule" : { "minute" : [30], "hour" : [8,17] }`

Deep Dive: *startTime*

The following table captures how *startTime* controls how a job is run.

STARTTIME VALUE	NO RECURRENCE	RECURRENCE. NO SCHEDULE	RECURRENCE WITH SCHEDULE
No start time	Run once immediately	Run once immediately. Run subsequent executions based on calculating from last execution time	Run once immediately Run subsequent executions based on recurrence schedule
Start time in past	Run once immediately	Calculate first future execution time after start time, and run at that time Run subsequent executions based oncalculating from last execution time See example after this table for a further explanation	Job starts *no sooner than* the specified start time. The first occurrence is based on the schedule calculated from the start time Run subsequent executions based on recurrence schedule
Start time in future or at present	Run once at specified start time	Run once at specified start time Run subsequent executions based on calculating from last execution time	Job starts *no sooner than* the specified start time. The first occurrence is based on the schedule calculated from the start time Run subsequent executions based on recurrence schedule

Let's see an example of what happens where *startTime* is in the past, with *recurrence* but no *schedule*. Assume that the current time is 2015-04-08 13:00, *startTime* is 2015-04-07 14:00, and *recurrence* is every 2 days (defined with *frequency*: day and *interval*: 2.) Note that the *startTime* is in the past, and occurs before the current time

Under these conditions, the *first execution* will be 2015-04-09 at 14:00. The Scheduler engine calculates execution occurrences from the start time. Any instances in the past are discarded. The engine uses the next instance that occurs in the future. So in this case, *startTime* is 2015-04-07 at 2:00pm, so the next instance is 2 days from that time, which is 2015-04-09 at 2:00pm.

Note that the first execution would be the same even if the startTime 2015-04-05 14:00 or 2015-04-01 14:00. After the first execution, subsequent executions are calculated using the scheduled – so they'd be at 2015-04-11 at 2:00pm, then 2015-04-13 at 2:00pm, then 2015-04-15 at 2:00pm, etc.

Finally, when a job has a schedule, if hours and/or minutes aren't set in the schedule, they default to the hours and/or minutes of the first execution, respectively.

Deep Dive: *schedule*

On one hand, a *schedule* can *limit* the number of job executions. For example, if a job with a "month" frequency has a *schedule* that runs on only day 31, the job runs in only those months that have a 31st day.

On the other hand, a *schedule* can also *expand* the number of job executions. For example, if a job with a "month" frequency has a *schedule* that runs on month days 1 and 2, the job runs on the 1st and 2nd days of the month instead of just once a month.

If multiple schedule elements are specified, the order of evaluation is from the largest to smallest – week number, month day, week day, hour, and minute.

The following table describes *schedule* elements in detail.

JSON NAME	DESCRIPTION	VALID VALUES
minutes	Minutes of the hour at which the job will run	• Integer, or • Array of integers
hours	Hours of the day at which the job will run	• Integer, or • Array of integers
weekDays	Days of the week the job will run. Can only be specified with a weekly frequency.	• "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", or "Sunday" • Array of any of the above values (max array size 7) *Not* case-sensitive
monthlyOccurrences	Determines which days of the month the job will run. Can only be specified with a monthly frequency.	• Array of monthlyOccurrence objects: `{ "day": *day*, "occurrence": *occurrence* }` *day* is the day of the week the job will run, e.g. {Sunday} is every Sunday of the month. Required. Occurrence is *occurrence* of the day during the month, e.g. {Sunday, -1} is the last Sunday of the month. Optional.
monthDays	Day of the month the job will run. Can only be specified with a monthly frequency.	• Any value <= -1 and >= -31. • Any value >= 1 and <= 31. • An array of above values

Examples: Recurrence Schedules

The following are various examples of recurrence schedules – focusing on the schedule object and its sub-elements.

The schedules below all assume that the *interval* is set to 1. Also, one must assume the right frequency in accordance to what is in the *schedule* – e.g., one can't use frequency "day" and have a "monthDays" modification in the schedule. Such restrictions are described above.

EXAMPLE	DESCRIPTION

EXAMPLE	DESCRIPTION
`{"hours":[5]}`	Run at 5AM Every Day. Azure Scheduler matches up each value in "hours" with each value in "minutes", one by one, to create a list of all the times at which the job is to be run.
`{"minutes":[15], "hours":[5]}`	Run at 5:15AM Every Day
`{"minutes":[15], "hours":[5,17]}`	Run at 5:15 AM and 5:15 PM Every Day
`{"minutes":[15,45], "hours":[5,17]}`	Run at 5:15AM, 5:45AM, 5:15PM, and 5:45PM Every Day
`{"minutes":[0,15,30,45]}`	Run Every 15 Minutes
`{hours":[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]}`	Run Every Hour. This job runs every hour. The minute is controlled by the *startTime*, if one is specified, or if none is specified, by the creation time. For example, if the start time or creation time (whichever applies) is 12:25 PM, the job will be run at 00:25, 01:25, 02:25, …, 23:25. The schedule is equivalent to having a job with *frequency* of "hour", an *interval* of 1, and no *schedule*. The difference is that this schedule could be used with different *frequency* and *interval* to create other jobs too. For example, if the *frequency* were "month", the schedule would run only once a month instead of every day if *frequency* were "day"
`{minutes:[0]}`	Run Every Hour on the Hour. This job also runs every hour, but on the hour (e.g. 12AM, 1AM, 2AM, etc.) This is equivalent to a job with frequency of "hour", a startTime with zero minutes, and no schedule if the frequency were "day", but if the frequency were "week" or "month," the schedule would execute only one day a week or one day a month, respectively.
`{"minutes":[15]}`	Run at 15 Minutes Past Hour Every Hour. Runs every hour, starting at 00:15AM, 1:15AM, 2:15AM, etc. and ending at 10:15PM and 11:15PM.
`{"hours":[17], "weekDays":["saturday"]}`	Run at 5PM on Saturdays Every Week
`{hours":[17], "weekDays":["monday", "wednesday", "friday"]}`	Run at 5PM on Monday, Wednesday, and Friday Every Week
`{"minutes":[15,45], "hours":[17], "weekDays": ["monday", "wednesday", "friday"]}`	Run at 5:15PM and 5:45PM on Monday, Wednesday, and Friday Every Week
`{"hours":[5,17], "weekDays":["monday", "wednesday", "friday"]}`	Run at 5AM and 5PM on Monday, Wednesday, and Friday Every Week
`{"minutes":[15,45], "hours":[5,17], "weekDays": ["monday", "wednesday", "friday"]}`	Run at 5:15AM, 5:45AM, 5:15PM, and 5:45PM on Monday, Wednesday, and Friday Every Week
`{"minutes":[0,15,30,45], "weekDays":["monday", "tuesday", "wednesday", "thursday", "friday"]}`	Run Every 15 Minutes on Weekdays

EXAMPLE	DESCRIPTION
`{"minutes":[0,15,30,45], "hours": [9, 10, 11, 12, 13, 14, 15, 16] "weekDays":["monday", "tuesday", "wednesday", "thursday", "friday"]}`	Run Every 15 Minutes on Weekdays between 9AM and 4:45PM
`{"weekDays":["sunday"]}`	Run on Sundays at Start Time
`{"weekDays":["tuesday", "thursday"]}`	Run on Tuesdays and Thursdays at Start Time
`{"minutes":[0], "hours":[6], "monthDays":[28]}`	Run at 6AM on the 28th Day of Every Month (assuming frequency of month)
`{"minutes":[0], "hours":[6], "monthDays":[-1]}`	Run at 6AM on the Last Day of the Month. If you'd like to run a job on the last day of a month, use -1 instead of day 28, 29, 30, or 31.
`{"minutes":[0], "hours":[6], "monthDays":[1,-1]}`	Run at 6AM on the First and Last Day of Every Month
`{monthDays":[1,-1]}`	Run on the First and Last Day of Every Month at Start Time
`{monthDays":[1,14]}`	Run on the First and Fourteenth Day of Every Month at Start Time
`{monthDays":[2]}`	Run on the Second Day of the Month at Start Time
`{"minutes":[0], "hours":[5], "monthlyOccurrences": [{"day":"friday", "occurrence":1}]}`	Run on First Friday of Every Month at 5AM
`{"monthlyOccurrences":[{"day":"friday", "occurrence":1}]}`	: Run on First Friday of Every Month at Start Time
`{"monthlyOccurrences":[{"day":"friday", "occurrence":-3}]}`	Run on Third Friday from End of Month, Every Month, at Start Time
`{"minutes":[15], "hours":[5], "monthlyOccurrences": [{"day":"friday", "occurrence":1},{"day":"friday", "occurrence":-1}]}`	Run on First and Last Friday of Every Month at 5:15AM
`{"monthlyOccurrences":[{"day":"friday", "occurrence":1},{"day":"friday", "occurrence":-1}]}`	Run on First and Last Friday of Every Month at Start Time
`{"monthlyOccurrences":[{"day":"friday", "occurrence":5}]}`	Run on Fifth Friday of Every Month at Start Time. If there is no fifth Friday in a month, this does not run, since it's scheduled to run on only fifth Fridays. You may consider using -1 instead of 5 for the occurrence if you want to run the job on the last occurring Friday of the month.
`{"minutes":[0,15,30,45], "monthlyOccurrences": [{"day":"friday", "occurrence":-1}]}`	Run Every 15 Minutes on Last Friday of the Month
`{"minutes":[15,45], "hours":[5,17], "monthlyOccurrences":[{"day":"wednesday", "occurrence":3}]}`	Run at 5:15AM, 5:45AM, 5:15PM, and 5:45PM on the 3rd Wednesday of Every Month

See Also

Scheduler Outbound Authentication

6/27/2017 • 5 min to read • Edit Online

Scheduler jobs may need to call out to services that require authentication. This way, a called service can determine if the Scheduler job can access its resources. Some of these services include other Azure services, Salesforce.com, Facebook, and secure custom websites.

Adding and Removing Authentication

Adding authentication to a Scheduler job is simple – add a JSON child element `authentication` to the `request` element when creating or updating a job. Secrets passed to the Scheduler service in a PUT, PATCH, or POST request – as part of the `authentication` object – are never returned in responses. In responses, secret information is set to null or may have a public token that represents the authenticated entity.

To remove authentication, PUT or PATCH the job explicitly, setting the `authentication` object to null. You will not see any authentication properties back in response.

Currently, the only supported authentication types are the `ClientCertificate` model (for using the SSL/TLS client certificates), the `Basic` model (for Basic authentication), and the `ActiveDirectoryOAuth` model (for Active Directory OAuth authentication.)

Request Body for ClientCertificate Authentication

When adding authentication using the `ClientCertificate` model, specify the following additional elements in the request body.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using an SSL client certificate.
type	Required. Type of authentication.For SSL client certificates, the value must be `ClientCertificate`.
pfx	Required. Base64-encoded contents of the PFX file.
password	Required. Password to access the PFX file.

Response Body for ClientCertificate Authentication

When a request is sent with authentication info, the response contains the following authentication-related elements.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using an SSL client certificate.
type	Type of authentication. For SSL client certificates, the value is `ClientCertificate`.

ELEMENT	DESCRIPTION
certificateThumbprint	The thumbprint of the certificate.
certificateSubjectName	The subject distinguished name of the certificate.
certificateExpiration	The expiration date of the certificate.

Sample REST Request for ClientCertificate Authentication

```
PUT https://management.azure.com/subscriptions/1fe0abdf-581e-4dfe-9ec7-e5cb8e7b205e/resourceGroups/CS-
SoutheastAsia-scheduler/providers/Microsoft.Scheduler/jobcollections/southeastasiajc/jobs/httpjob?api-
version=2016-01-01 HTTP/1.1
User-Agent: Fiddler
Host: management.azure.com
Authorization: Bearer sometoken
Content-Type: application/json; charset=utf-8

{
  "properties": {
    "startTime": "2015-05-14T14:10:00Z",
    "action": {
      "request": {
        "uri": "https://mywebserviceendpoint.com",
        "method": "GET",
        "headers": {
          "x-ms-version": "2013-03-01"
        },
        "authentication": {
          "type": "clientcertificate",
          "password": "password",
          "pfx": "pfx key"
        }
      },
      "type": "http"
    },
    "recurrence": {
      "frequency": "minute",
      "endTime": "2016-04-10T08:00:00Z",
      "interval": 1
    },
    "state": "enabled",
  }
}
```

Sample REST Response for ClientCertificate Authentication

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 858
Content-Type: application/json; charset=utf-8
Expires: -1
x-ms-request-id: 56c7b40e-721a-437e-88e6-f68562a73aa8
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
x-ms-ratelimit-remaining-subscription-resource-requests: 599
x-ms-correlation-request-id: 1075219e-e879-4030-bc81-094e54fbabce
x-ms-routing-request-id: WESTUS:20160316T190424Z:1075219e-e879-4030-bc81-094e54fbabce
Strict-Transport-Security: max-age=31536000; includeSubDomains
Date: Wed, 16 Mar 2016 19:04:23 GMT

{
  "id": "/subscriptions/1fe0abdf-581e-4dfe-9ec7-e5cb8e7b205e/resourceGroups/CS-SoutheastAsia-
scheduler/providers/Microsoft.Scheduler/jobCollections/southeastasiajc/jobs/httpjob",
  "type": "Microsoft.Scheduler/jobCollections/jobs",
  "name": "southeastasiajc/httpjob",
  "properties": {
    "startTime": "2015-05-14T14:10:00Z",
    "action": {
      "request": {
        "uri": "https://mywebserviceendpoint.com",
        "method": "GET",
        "headers": {
          "x-ms-version": "2013-03-01"
        },
        "authentication": {
          "certificateThumbprint": "88105CG9DF9ADE75B835711D899296CB217D7055",
          "certificateExpiration": "2021-01-01T07:00:00Z",
          "certificateSubjectName": "CN=Scheduler Mgmt",
          "type": "ClientCertificate"
        }
      },
      "type": "http"
    },
    "recurrence": {
      "frequency": "minute",
      "endTime": "2016-04-10T08:00:00Z",
      "interval": 1
    },
    "state": "enabled",
    "status": {
      "nextExecutionTime": "2016-03-16T19:05:00Z",
      "executionCount": 0,
      "failureCount": 0,
      "faultedCount": 0
    }
  }
}
```

Request Body for Basic Authentication

When adding authentication using the `Basic` model, specify the following additional elements in the request body.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using Basic authentication.

ELEMENT	DESCRIPTION
type	Required. Type of authentication. For Basic authentication, the value must be `Basic`.
username	Required. Username to authenticate.
password	Required. Password to authenticate.

Response Body for Basic Authentication

When a request is sent with authentication info, the response contains the following authentication-related elements.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using Basic authentication.
type	Type of authentication. For Basic authentication, the value is `Basic`.
username	The authenticated username.

Sample REST Request for Basic Authentication

```
PUT https://management.azure.com/subscriptions/1d908808-e491-4fe5-b97e-29886e18efd4/resourceGroups/CS-SoutheastAsia-scheduler/providers/Microsoft.Scheduler/jobcollections/southeastasiajc/jobs/httpjob?api-version=2016-01-01 HTTP/1.1
User-Agent: Fiddler
Host: management.azure.com
Authorization: Bearer sometoken
Content-Length: 562
Content-Type: application/json; charset=utf-8

{
  "properties": {
    "startTime": "2015-05-14T14:10:00Z",
    "action": {
      "request": {
        "uri": "https://mywebserviceendpoint.com",
        "method": "GET",
        "headers": {
          "x-ms-version": "2013-03-01"
        },
        "authentication": {
          "type": "basic",
          "username": "user",
          "password": "password"
        }
      },
      "type": "http"
    },
    "recurrence": {
      "frequency": "minute",
      "endTime": "2016-04-10T08:00:00Z",
      "interval": 1
    },
    "state": "enabled",
  }
}
```

Sample REST Response for Basic Authentication

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 701
Content-Type: application/json; charset=utf-8
Expires: -1
x-ms-request-id: a2dcb9cd-1aea-4887-8893-d81273a8cf04
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
x-ms-ratelimit-remaining-subscription-resource-requests: 599
x-ms-correlation-request-id: 7816f222-6ea7-468d-b919-e6ddebbd7e95
x-ms-routing-request-id: WESTUS:20160316T190506Z:7816f222-6ea7-468d-b919-e6ddebbd7e95
Strict-Transport-Security: max-age=31536000; includeSubDomains
Date: Wed, 16 Mar 2016 19:05:06 GMT

{
   "id":"/subscriptions/1d908808-e491-4fe5-b97e-29886e18efd4/resourceGroups/CS-SoutheastAsia-
scheduler/providers/Microsoft.Scheduler/jobCollections/southeastasiajc/jobs/httpjob",
   "type":"Microsoft.Scheduler/jobCollections/jobs",
   "name":"southeastasiajc/httpjob",
   "properties":{
      "startTime":"2015-05-14T14:10:00Z",
      "action":{
         "request":{
            "uri":"https://mywebserviceendpoint.com",
            "method":"GET",
            "headers":{
               "x-ms-version":"2013-03-01"
            },
            "authentication":{
               "username":"user1",
               "type":"Basic"
            }
         },
         "type":"http"
      },
      "recurrence":{
         "frequency":"minute",
         "endTime":"2016-04-10T08:00:00Z",
         "interval":1
      },
      "state":"enabled",
      "status":{
         "nextExecutionTime":"2016-03-16T19:06:00Z",
         "executionCount":0,
         "failureCount":0,
         "faultedCount":0
      }
   }
}
```

Request Body for ActiveDirectoryOAuth Authentication

When adding authentication using the `ActiveDirectoryOAuth` model, specify the following additional elements in the request body.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using ActiveDirectoryOAuth authentication.

ELEMENT	DESCRIPTION
type	Required. Type of authentication. For ActiveDirectoryOAuth authentication, the value must be `ActiveDirectoryOAuth`.
tenant	Required. The tenant identifier for the Azure AD tenant.
audience	Required. This is set to https://management.core.windows.net/.
clientId	Required. Provide the client identifier for the Azure AD application.
secret	Required. Secret of the client that is requesting the token.

Determining your Tenant Identifier

You can find the tenant identifier for the Azure AD tenant by running `Get-AzureAccount` in Azure PowerShell.

Response Body for ActiveDirectoryOAuth Authentication

When a request is sent with authentication info, the response contains the following authentication-related elements.

ELEMENT	DESCRIPTION
authentication (parent element)	Authentication object for using ActiveDirectoryOAuth authentication.
type	Type of authentication. For ActiveDirectoryOAuth authentication, the value is `ActiveDirectoryOAuth`.
tenant	The tenant identifier for the Azure AD tenant.
audience	This is set to https://management.core.windows.net/.
clientId	The client identifier for the Azure AD application.

Sample REST Request for ActiveDirectoryOAuth Authentication

```
PUT https://management.azure.com/subscriptions/1d908808-e491-4fe5-b97e-29886e18efd4/resourceGroups/CS-SoutheastAsia-scheduler/providers/Microsoft.Scheduler/jobcollections/southeastasiajc/jobs/httpjob?api-version=2016-01-01 HTTP/1.1
User-Agent: Fiddler
Host: management.azure.com
Authorization: Bearer sometoken
Content-Length: 757
Content-Type: application/json; charset=utf-8

{
  "properties": {
    "startTime": "2015-05-14T14:10:00Z",
    "action": {
      "request": {
        "uri": "https://mywebserviceendpoint.com",
        "method": "GET",
        "headers": {
          "x-ms-version": "2013-03-01"
        },
        "authentication": {
          "tenant":"microsoft.onmicrosoft.com",
          "audience":"https://management.core.windows.net/",
          "clientId":"dc23e764-9be6-4a33-9b9a-c46e36f0c137",
          "secret": "G6u071r8Gjw4V4KSibnb+VK4+tX399hkHaj7LOyHuj5=",
          "type":"ActiveDirectoryOAuth"
        }
      },
      "type": "http"
    },
    "recurrence": {
      "frequency": "minute",
      "endTime": "2016-04-10T08:00:00Z",
      "interval": 1
    },
    "state": "enabled",
  }
}
```

Sample REST Response for ActiveDirectoryOAuth Authentication

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 885
Content-Type: application/json; charset=utf-8
Expires: -1
x-ms-request-id: 86d8e9fd-ac0d-4bed-9420-9baba1af3251
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
x-ms-ratelimit-remaining-subscription-resource-requests: 599
x-ms-correlation-request-id: 5183bbf4-9fa1-44bb-98c6-6872e3f2e7ce
x-ms-routing-request-id: WESTUS:20160316T191003Z:5183bbf4-9fa1-44bb-98c6-6872e3f2e7ce
Strict-Transport-Security: max-age=31536000; includeSubDomains
Date: Wed, 16 Mar 2016 19:10:02 GMT

{
   "id":"/subscriptions/1d908808-e491-4fe5-b97e-29886e18efd4/resourceGroups/CS-SoutheastAsia-
scheduler/providers/Microsoft.Scheduler/jobCollections/southeastasiajc/jobs/httpjob",
   "type":"Microsoft.Scheduler/jobCollections/jobs",
   "name":"southeastasiajc/httpjob",
   "properties":{
      "startTime":"2015-05-14T14:10:00Z",
      "action":{
         "request":{
            "uri":"https://mywebserviceendpoint.com",
            "method":"GET",
            "headers":{
               "x-ms-version":"2013-03-01"
            },
            "authentication":{
               "tenant":"microsoft.onmicrosoft.com",
               "audience":"https://management.core.windows.net/",
               "clientId":"dc23e764-9be6-4a33-9b9a-c46e36f0c137",
               "type":"ActiveDirectoryOAuth"
            }
         },
         "type":"http"
      },
      "recurrence":{
         "frequency":"minute",
         "endTime":"2016-04-10T08:00:00Z",
         "interval":1
      },
      "state":"enabled",
      "status":{
         "lastExecutionTime":"2016-03-16T19:10:00.3762123Z",
         "nextExecutionTime":"2016-03-16T19:11:00Z",
         "executionCount":5,
         "failureCount":5,
         "faultedCount":1
      }
   }
}
```

See Also

What is Scheduler?

Azure Scheduler concepts, terminology, and entity hierarchy

Get started using Scheduler in the Azure portal

Plans and billing in Azure Scheduler

Azure Scheduler REST API reference

Azure Scheduler PowerShell cmdlets reference

Azure Scheduler high-availability and reliability

Azure Scheduler limits, defaults, and error codes